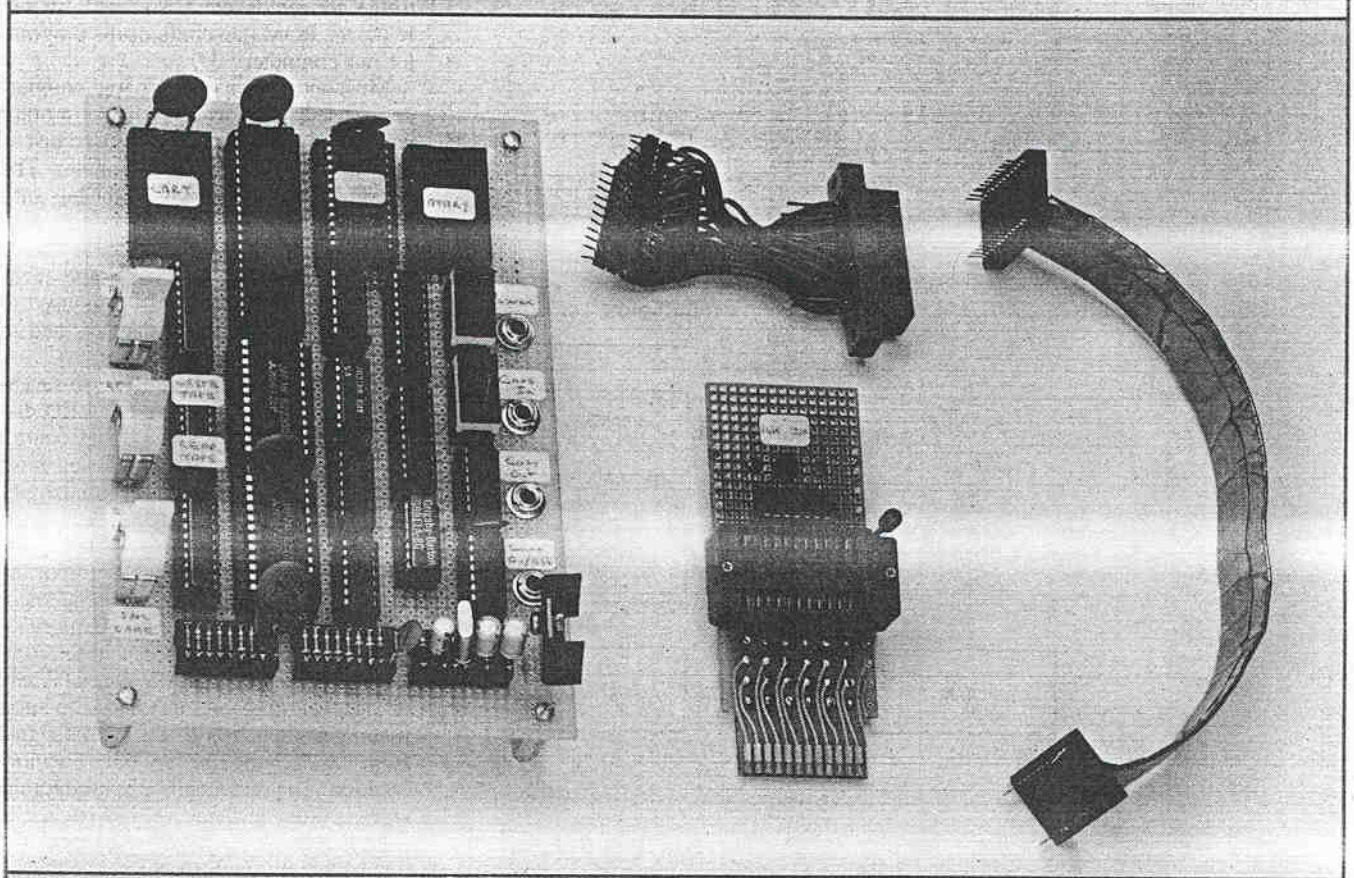
ATARI Game Recorder

GUY VACHON and DAVID A. CHAN



Store your library of Atari videogame cartridges on cassette tapes!

DID YOU EVER WISH THAT YOU COULD make copies of game cartridges for your Atari *2600*? Well, with the circuit we'll describe, you can! We'll show you how to record the contents of your cartridges on cassette tape—and how to load the game back into the *2600*. Before we get into the details of the circuit, let's review some basics.

As you probably know, a videogame is just a simplified home computer—one that has been dedicated to the specific task of playing games. Keep in mind, though, that the videogame operates much the same as any computer—the electronic circuits that make up the machine (the hardware) execute instructions that make up the game (the software).

The software is stored, of course, in the game cartridge, which consists simply of ROM (*R*ead-*O*nly *M*emory). As its name implies, you cannot change the contents of ROM—that type of memory can be read but not written to.

If a videogame is just a simple home computer, as we stated earlier, you might wonder why the programs stored in the ROM cartridges cannot be stored on magnetic tape or floppy disks like programs for other home computers. Well, they can! But videogames like the Atari *2600* lack the necessary hardware to record them. And that's what this article is all about.

The basic approach

Our approach will be to copy the contents of the ROM cartridge into RAM. (That's *R*andom-*A*ccess *M*emory, also known as read/write memory.) Once we have the game program in RAM, we can then copy it to cassette tape.

Figure 1 shows a very basic block diagram of what we need: the videogame, a game cartridge (ROM), a cassette player, RAM, and a control device, which allows us to correctly direct the flow of data. For example, the first step in copying a game cartridge is to load the contents into RAM. Then the contents of RAM is transferred to cassette tape much in the same way that many home computers save programs on cassette. (The Timex Sinclair *1000*, is one example.) When we want to play the game, we reload it from cassette to RAM. We get the *2600* to think that the RAM is just a game-cartridge ROM by setting the READ/WRITE input of the RAM to READ and connecting its other inputs and outputs to the *2600* just as if it were ROM.

We can also make tape-to-tape copies easily using that scheme. Once we load the program from cassette into RAM, we can simply dump the RAM contents to another tape.

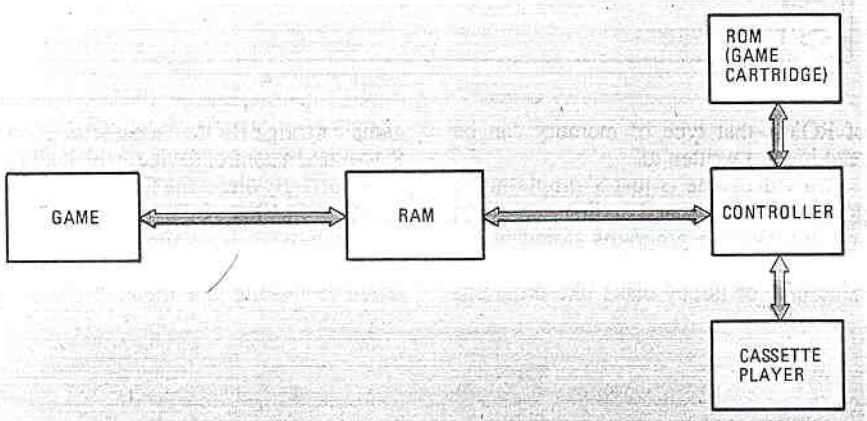


FIG. 1—A CONTROLLER IS NEEDED to properly direct the flow of data from the game cartridge ROM to the computer's RAM, from the cassette player to the RAM, etc.

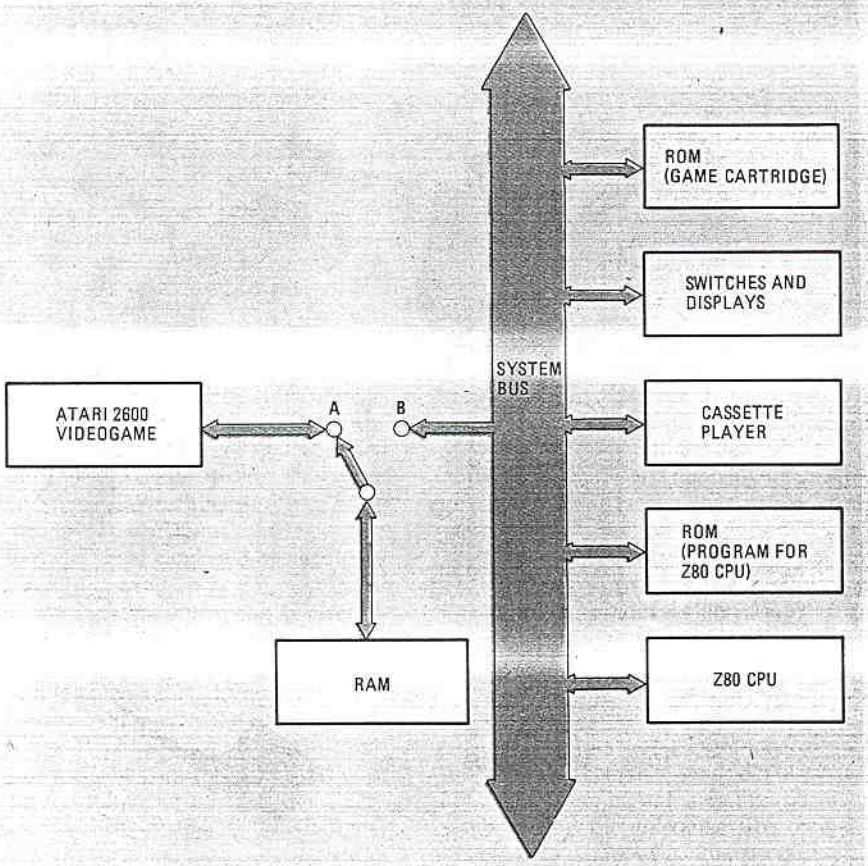


FIG. 2—A DEDICATED, SIMPLIFIED COMPUTER. This block diagram gives a basic idea of what we need to record the contents of Atari *2600* videogame cartridges.

A dedicated computer

If you're familiar with home computers—even the cheapest models that you can buy for under $50—you know that they have all the capabilities we need. We could approach the problem by modifying a computer to do exactly what we want. But that is not the way we will go. Instead, we will build our own dedicated, simplified computer.

A block diagram of the computer that we need is shown in Fig. 2. When the switch is in position "B," the *2600* is out of the picture and we're left with only our dedicated computer. It sees the game cartridge and RAM as part of its memory. It can transfer data from the ROM cartridge to the RAM, and it can store and retrieve programs from tape. The game cartridge is not the only ROM: Another block of ROM holds what can be thought of as the operating system of our computer. It contains the instructions that tell the Z80 CPU how to perform the appropriate data-transfer tasks.

Note that also "hanging from the bus" of our computer are switches and displays that are used for I/O. By setting the switches, we can give the computer certain commands. The displays let the computer tell us what it is doing.

When the switch shown in Fig. 2 is moved to position "A," the Atari *2600* videogame uses the RAM simply as if it were ROM. So, for example, after you loaded the RAM with a program contained from cassette, you would flip the switch to position "A" so that the *2600* could see it.

Game-recorder computer hardware

Let's look at the hardware that we'll use to help us record game cartridges. Figure 3 shows the schematic of the computer/recorder. As you can see, the computer is structured around the Z80 bus. Connected to the bus, directly or through buffers, are all the computer's components: the Z80 microprocessor (IC3), the RAM (IC11–IC13), and the I/O devices (S1–S5, DISP1, DISP2, cassette output, game cartridge connector, etc.). We can also see IC10, the ROM that contains the program for our computer.

Most of the components of our computer are used in the usual fashion. In other words, the ROM and RAM is used just as it is in any given home computer. The cartridge connects directly to the bus and, as far as our computer is concerned, seems to be another several kilobytes of addressable memory. That same technique of memory-mapped I/O is used to drive the seven-segment displays and to interface with the tape recorder.

Note that we do not use BCD-to-seven-segment decoders to drive our LED displays. Instead, the Z80 CPU has control over the segments and turns them on or off as needed to represent hexadecimal digits Ø–F and an error message of three horizontal bars. But we're getting a little ahead of ourselves. What is important here is that, as far as the Z80 is concerned, the displays are "write only" memory locations. Information encoded in the common seven-segment display format is sent to the display at their locations. The information is latched with the WRITE signal from the Z80 just like any other memory location. The same method is used for the tape-recorder interface. A 1 is latched to send a high-level voltage to the tape and a Ø is latched to send a low-level voltage.

The last major component of the computer is the interface to the Atari *2600*. That interface is essentially made up of IC4, IC5, and IC6—three 74LS244 octal buffers with three-state outputs. By looking at the direction of the buffers, you can see that the dedicated computer accepts addresses from the Atari game and outputs data to it. (Remember: That's just what ROM does!) The buffers are enabled by the BUS ACKNOWLEDGE signal (pin 23) from the Z80 (IC3).

When we want the *2600* to play a game, we simply close the SETUP/PLAY switch, S6, which brings the Z80's BUS REQUEST line (pin 25) low. Thus, the *2600* actually does DMA (*D*irect *M*emory *A*ccess) on the computer when requested by you through the SETUP/PLAY switch.

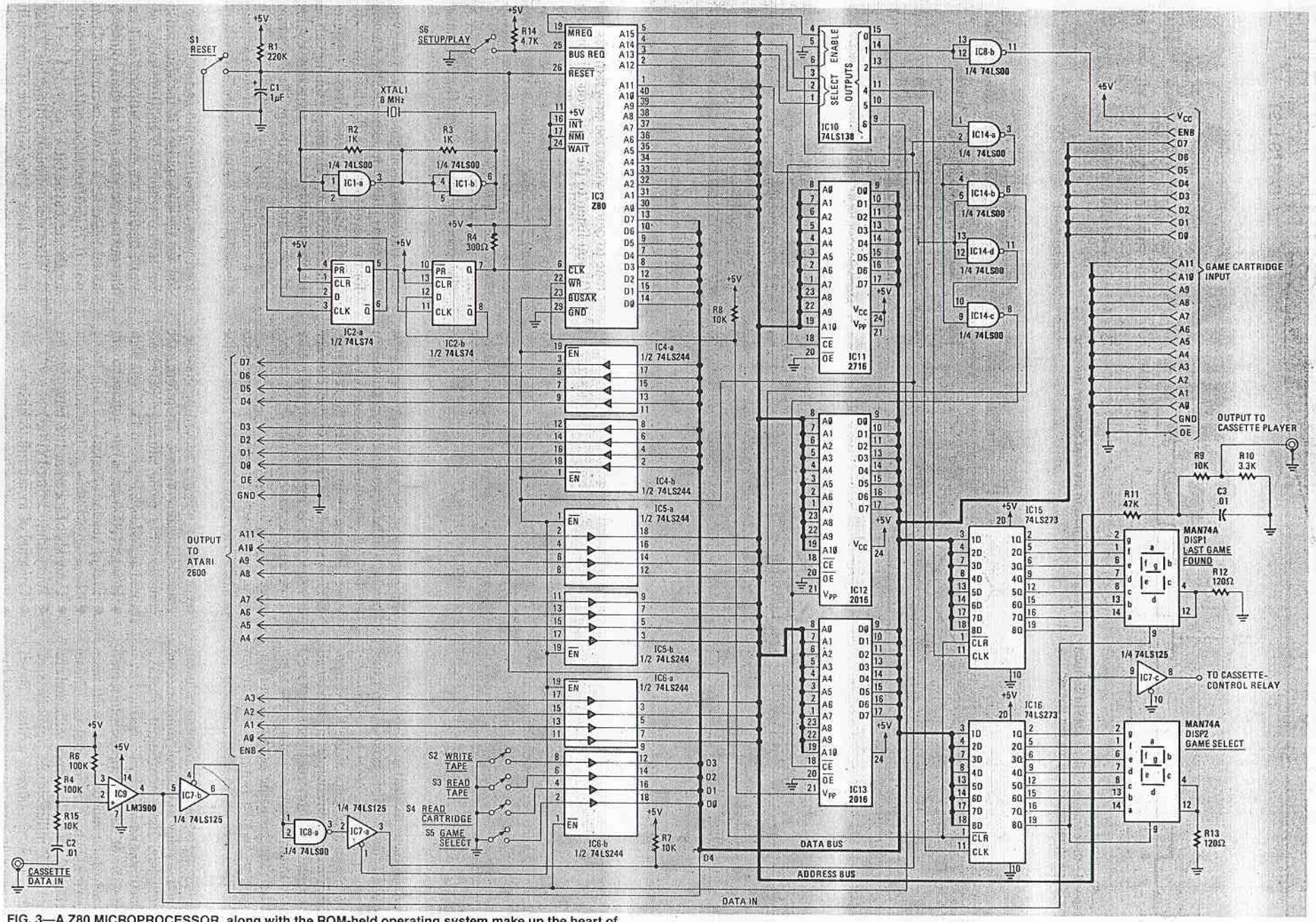


FIG. 3—A Z80 MICROPROCESSOR, along with the ROM-held operating system make up the heart of our computer/recorder. The game-cartridge ROM, LED displays, and cassette input/output are memory mapped.

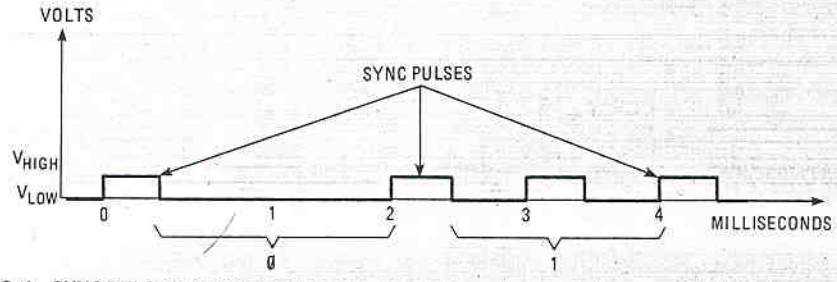


FIG. 4—SYNC PULSES, 2 milliseconds apart, are used to make sure that—even with slight changes in tape speed—the computer will be able to correctly read a tape. Data pulses are sent between the sync pulses: the sequence "Ø1" is shown above.

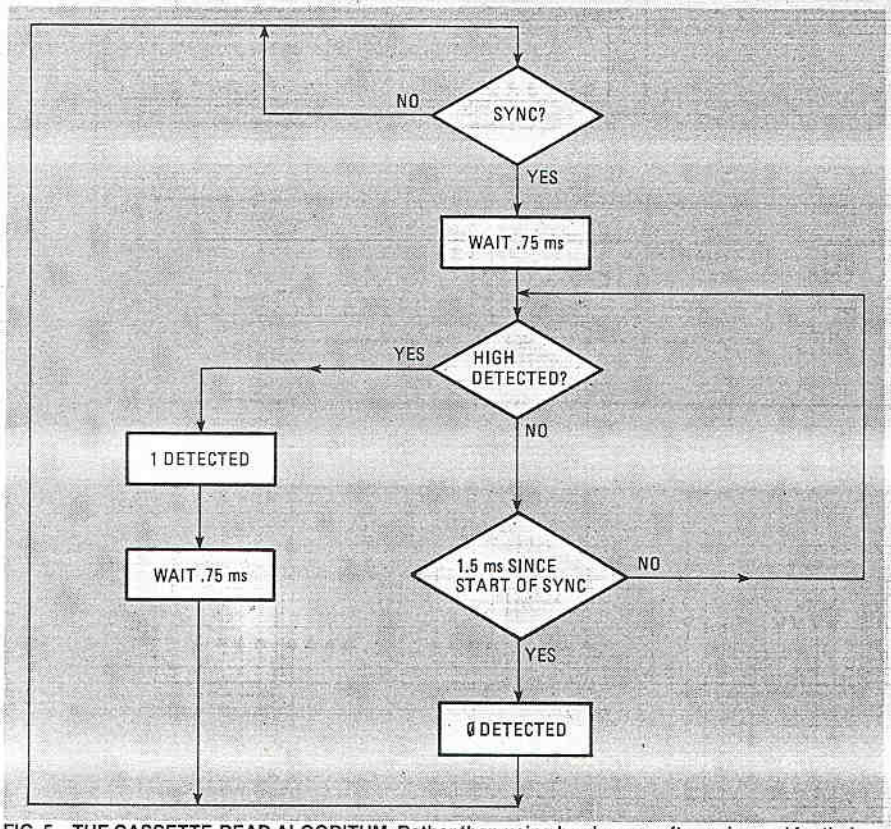


FIG. 5—THE CASSETTE-READ ALGORITHM. Rather than using hardware, software is used for timing operations.

PARTS LIST

All resistors are ¼-watt, 5%, unless otherwise specified.
R1—220,000 ohms
R2, R3—1000 ohms
R4—330 ohms
R5, R7–R9—10,000 ohms
R6, R15—100,000 ohms
R10—3300 ohms
R11—47,000 ohms
R12, R13—120 ohms
R14—4700 ohms
Capacitors
C1—1 μF, 10 volts, electrolytic
C2, C3—.01 μF, ceramic disc
Semiconductors
IC1, IC8, IC14—74LS00 quad 2-input NAND gate
IC2—74LS74 dual D-type flip-flop
IC3—Z80 microprocessor
IC4–IC6—74LS244 octal buffer
IC7—74LS125 quad bus buffer
IC9—LM3900 quad op-amp
IC10—74LS138 3-to-8 line decoder
IC11—2716 EPROM containing the computer's operating system
IC12, IC13—2016 2K × 8 static RAM
IC15, IC16—74LS273 octal D-type flip-flop
DISP1, DISP2—MAN74A
S1–S6—SPST switches
XTAL—8 MHz

Cassette input/output

Reading the ROM, of course, isn't the only job of our computer/recorder—we must write the contents of the ROM into tape. We'll do that by outputting one bit at a time by sending different voltage waveforms to the microphone input of the tape recorder. The bit to be output is put on data line D8, which is stored in IC16 (a 74LS273 D-type flip-flop), which is clocked by the $\overline{MREQ}$ line of the Z80. (Note that the output of IC7-c can be used to switch a relay to control your cassette player through its REMOTE input. That is, of course, optional.)

We will not only send pulses to represent zero bits and one bits—we'll also send out synchronization pulses. Those pulses are 0.25 milliseconds wide and are sent every 2 milliseconds, regardless of whether a 1 or 0 is being written. Those sync pulses are used to ensure that if the tape recorder speed varies slightly, our computer will be able to keep track. A data bit will be represented by a pulse—or the lack of a pulse—between the sync pulses. Figure 4 shows what the sequence "Ø1" would look like. As we can see there, a "Ø" is represented by no pulse between sync pulses, and a "1" is represented by a pulse.

Each instruction for the *2600* consists of 8 bits. We will add a parity bit to be able to detect if a program has been misrecorded or when a recording has degraded and has errors. Therefore, the contents of a game-cartridge ROM are stored as follows:

- Header (2000 zeros)
- End of header (4 ones)
- Name tag (4 bits)
- Contents of location 1 (8 bits)
- Parity for location 1 (1 bit)
- Contents of location 2 (8 bits)
- Parity for location 2 (1 bits)

That continues until all the ROM's contents are stored.

The header serves two purposes: It separates programs and provides an audible tone to detect where the program begins. (If you listen to the tape, you will hear a high pitch tone for the header. The program itself sounds like high- and low-noise.) The name tag allows you to save several programs on one cassette, and to search for those programs.

Reading the ROM contents from tape is also done one bit at a time. The computer/recorder constantly monitors what is coming out of the tape and can tell whenever the output is high or low. The sync (and data) pulses are detected by waiting for the level to go from low to high.

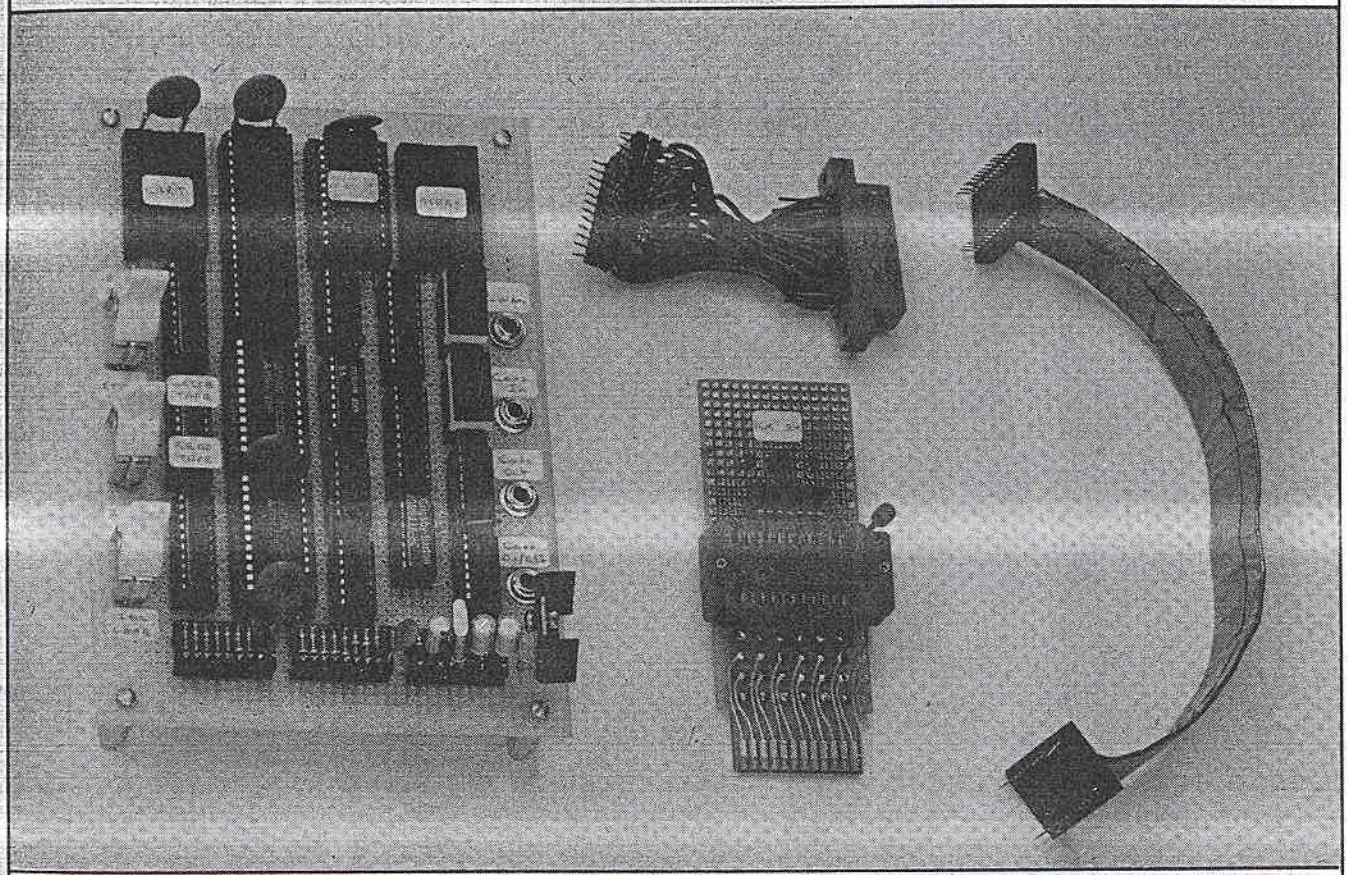
Detecting those pulses doesn't require much hardware. The proper timing can be implemented by counting machine cycles in a loop that does nothing. The algorithm is illustrated by the flowchart in Fig. 5.

As you can see from the flowchart, the algorithm for cassette operation is very simple: Wait for the sync pulse to appear then wait .75 millisecond and start looking for the data pulse. If the data pulse isn't seen within 1.5 milliseconds, assume that a Ø was recorded, and wait for the next sync pulse. If a data pulse *is* found, assume a 1 was recorded. Then wait 0.75 milliseconds and start looking for the next sync bit.

When we continue next time, we'll take a closer look at the software. Then we'll give you some construction hints. **R-E**

ATARI Game Recorder

GUY VACHON and DAVID A. CHAN



You can record the contents of your Atari 208 videogame cartridges on audio cassette tape! This month, we'll take a look at the software that's needed.

Part 2 LAST MONTH, WE looked at the basic approach we'll follow to store the contents of Atari *2600* game cartridges on audio cassette tape. We also looked at the hardware that's required, and briefly studied how cassette I/O is handled. This time, we'll look at the software in more depth. Then we'll see how we can build the game recorder and put it to use.

Game-recorder software

The complete software listing for the game recorder's operating system appears in Table 1. Note that it is written in Z80 mnemonics. Although we won't be discussing the software line by line, you might want to study Table 1 to get the details.

When we left off last time, we were discussing cassette I/O. Figure 5 showed a flowchart that described the cassette-read algorithm. Let's look at the software in more detail to see how it's used to detect the data and sync pulses. (Remember that sync pulses are sent out every 2 milliseconds. Data pulses are sent between the sync pulses—a pulse represents a 1 bit, while the lack of a pulse represents a zero bit.)

When the contents of a game cartridge is written to a cassette tape, a header of 2000 zero bits preceeds the actual beginning of the program bits. After the header, the game recorder also writes a (user-selected) label before each game. When the game recorder reads the contents of a cassette tape, its software looks for fifty consecutive zeros to decide that it has found the beginning of a game program. You might wonder why we write 2000 zero bits and look for only 50. There's a very practical reason: It allows the automatic gain control (AGC) of most recorders enough time to settle down.

After the recorder finds 50 consecutive zero bits, it keeps on looking until it finds a 1 bit. It then checks the name tag, which is output to the LAST GAME FOUND display. If the name tag matches the name of the game you selected, it keeps on reading bytes and storing them in the RAM. If the tag doesn't match, the game recorder keeps looking for another start-of-game header. (We'll give more details on that—and other operation aspects of the computer—a little later on in this article.)

You may recall that a parity bit is added to each instruction so that the game recorder will recognize when something

TABLE 1—GAME-RECORDER SOFTWARE

```
CURRENT_GAME_OUT                EQU  8000H
GAME_SEL_OUT                    EQU  0A000H
INPUT                           EQU  0C000H
RAM_FIRST_BYTE_ADD              EQU  4000H
ROM_FIRST_BYTE_ADD              EQU  2000H
LAST_PLUS_1_BYTE_RAM_HIGH       EQU  50H
LAST_BYTE_ADD_RAM               EQU  4FFFH
;START OF INITIALIZATION
          .Z80
          LD   A,00H            ;CLEAR LEDS
          EXX
          LD   B,A
          LD   (CURRENT_GAME_OUT),A
          LD   A,3FH
          LD   C,A
          LD   (GAME_SEL_OUT),A
          EXX
;START OF MAIN PROGRAM
LOOP1:    LD   A,(INPUT)        ;SEE IF INC GAME
                                ; PUSHED
          AND  01H
          JP   NZ, LAB1
          LD   A,0FFH           ;WAIT AND CHECK
                                ; INPUT AGAIN
LOOPA:    DEC  A
          JP   NZ,LOOPA
          LD   A,(INPUT)
          JP   NZ,LAB1
          EXX                   ;START OF INC GAME
                                ; PROGRAM
          LD   A,B
          INC  A
          AND  0FH
          LD   B,A
          EXX
          LD   IY,XX1
          JP   CONVERT
XX1:      EXX
          LD   C,A
          EXX
          LD   (GAME_SEL_OUT)A,
          LD   DE,7FFFH         ;WAIT HALF A SECOND
LOOPB:    DEC  A
          LD   A,E
          ADD  A,D
          JP   NZ,LOOPB
          JP   C,LOOPB
;CONTINUATION OF MAIN PROGRAM
LAB1:     LD   A,(INPUT)        ;SEE IF COPY PUSHED
          AND  02H
          JP   NZ,LAB2
          LD   A,0FFH           ;WAIT AND CHECK
                                ; INPUT AGAIN
LOOPC:    DEC  A
          JP   NZ,LOOPC
          LD   A,(INPUT)
          AND  02H
          JP   NZ,LAB2
          LD   DE,RAM_FIRST_BYTE_ADD
                                ;START OF COPY
                                ; PROGRAM
          LD   HL,ROM_FIRST_BYTE_ADD
          LD   BC,1000H
          LDIR
;CONTINUATION OF MAIN PROGRAM
LAB2:     LD   A,(INPUT)        ;SEE IF DOWNLOAD
                                ; PUSHED
          AND  04H
          JP   NZ,LAB3
          LD   A,0FFH           ;WAIT AND CHECK
                                ; INPUT AGAIN
LOOPD:    DEC  A
          JP   NZ,LOOPD
          LD   A,(INPUT)
          AND  04H
          JP   NZ,LAB3
          EXX                   ;START OF DOWNLOAD
                                ; PROGRAM
          LD   A,C              ;TURN ON DECIMAL
                                ; POINT
          EXX
          OR   80H
          LD   (GAME_SEL_OUT),A
BLOCIN:   LD   DE,01F4H
LOOPE:    LD   HL,XX2           ;FIND 500 ZEROS
          JP   BITIN
XX2:      JP   C,BLOCIN
          DEC  DE
          LD   A,E
          ADD  A,D
          JP   NZ,LOOPE
          JP   C,LOOPE
LOOPF:    LD   LH,XX3           ;FIND 1ST BIT OF 1ST
                                ; BYTE
          JP   BITIN
XX3;      HP   NC,LOOPF
          LD   A,01H            ;GET 1ST BYTE
          LD   B,07H
          LD   IX,XX4
          JP   BYTEIN
XX4;      AND  0FH              ;CONVERT TO 7 SEG
                                ; AND DISPLAY
          LD   IY,XX5
          JP   CONVERT
XX5:      LD   (CURRENT_GAME_OUT),A
          LD   D,A              ;SEE IF CORRECT
                                ; GAME
          EXX
          LD   A,C
          EXX
          CP   D
          JP   NZ,BLOCIN
          JD   DE,RAM_FIRST_BYTE_ADD
                                ;GET REST OF BLOCK
LOOPG:    LD   A,00H
          LD   B,08H
          LD   IX,XX6
          JP   BYTEIN
XX6:      LD   (DE),A
          INC  DE
          LD   A,D
          CP   LAST_PLUS_1_BYTE_RAM_HIGH
          JP   NZ,LOOPG
          LD   DE,0FFFFH        ;WAIT ONE SECOND
LOOPH;    DEC  DE
          LD   A,E
          ADD  A,D
          JP   NZ,LOOPH
          JP   C,LOOPH
          EXX                   ;TURN OFF DECIMAL
                                ; POINT
          LD   A,C
          EXX
          LD   (GAME_SEL_OUT),A
;CONTINUATION OF MAIN PROGRAM
LAB3:     LD   A,(INPUT)        ;SEE IF RECORD
                                ; PUSHED
          AND  08H
          JP   NZ,LAB4
          LD   A,0FFH           ;WAIT AND CHECK
                                ; AGAIN
LOOPI;    DEC  A
```

TABLE 1 (continued)

```
        JP    NZ,LOOPI
        LD    A,(INPUT)
        AND   08H
        JP    NZ,LAB4
        EXX                       ;START OF RECORD
                                  ; PROGRAM
        LD    A,C                 ;TURN ON DECIMAL
                                  ; POINT
        EXX
        OR    80H
        LD    (GAME_SEL_OUT),A
        LD    DE,07D0H
LOOPJ:  AND   0FFH                ;OUTPUT 2000 ZEROS
        LD    HL,XX7
        JP    BITOUT
XX7:    DEC   DE
        LD    A,E
        ADD   A,D
        JP    NZ,LOOPJ
        JP    C,LOOPJ
        EXX                       ;OUTPUT BLOCK
                                  ; ADDRESS
        LD    A,B
        EXX
        OR    0F0H
        LD    IX,XX8
        JP    BYTEOUT
XX8:    LD    DE,RAM_FIRST_BYTE_ADD
                                  ;OUTPUT BLOCK
LOOPK:  LD    A,(DE)
        LD    IX,XX9
        JP    BYTEOUT
XX9:    INC   DE
        CP    LAST_PLUS_1_BYTE_RAM_HIGH
        JP    NZ,LOOPK
        LD    DE,0FFFFH           ;WAIT ONE SECOND
LOOPL:  DEC   DE
        LD    A,E
        ADD   A,D
        JP    NZ,LOOPL
        JP    C,LOOPL
        EXX                       ;TURN OFF DECIMAL
                                  ; POINT
        LD    A,C
        EXX
        LD    (GAME_SEL_OUT),A
LAB4:   JP    LOOP1

;START OF SUBROUTINES
;BYTEIN SUBROUTINE - GETS ONE BYTE FROM TAPE
;IX = RETURN ADDRESS
;GIVEN: A IS (EMPTY) BYTE
;       B IS # BITS LEFT
;RESULT: A IS BYTE
;USES: B
;CALLS: BITIN
;CANNOT AFFECT: DE
BYTEIN: LD    HL,XX10             ;START OF BYTEIN
                                  ; PROGRAM
        JP    BITIN               ;GET ENTIRE BYTE
XX10:   RLA
        DEC   B
        JP    NZ,BYTEIN
        AND   0FFH                ;COMPUTE PARITY
        EX    AF,AF'
        LD    HL,XX11             ;GET PARITY BIT
        JP    BITIN
XX11:   JP    C,LAB5              ;SEE IF ERROR BY
                                  ; CHECKING 'CARRY'
                                  ; THEN
        EX    AF,AF'
        JP    PE,PERROR
        JP    (IX)
LAB5:   EX    AF,AF'              ;RECALL BYTE &
                                  ; FLAGS AND CHECK
                                  ; PARITY
        JP    PO,PERROR
        JP    (IX)
PERROR: LD    A,94H               ;ERROR, DISPLAY
                                  ; MESSAGE
        LD    (CURRENT_GAME_OUT),A
        LD    DE,LAST_BYTE_ADD_RAM
        JP    (IX)

;BITIN SUBROUTINE - GETS ONE BIT FROM TAPE
;HL = RETURN ADDRESS
;RESULT: BIT IS CARRY
;USES: D',E',H'
;CANNOT AFFECT: DE,B,A

BITIN:  EXX                       ;START OF BITIN
                                  ; PROGRAM
        LD    D,A                 ;EXCHANGE
                                  ; REGISTERS AND
                                  ; STORE A
LOOPM:  LD    A,(INPUT)           ;FIND HIGH
        AND   10H
        JP    Z,LOOPM
        LD    E,8FH               ;WAIT 1MSEC
LOOPN:  DEC   E
        JP    NZ,LOOPN
        LD    E,0CH               ;SEE IF 1 OR 0 FOR
                                  ; 0.25MSEC
LOOPO:  LD    A,(INPUT)
        AND   10H
        CCF
        JP    NZ,LAB6
        DEC   E
        JP    NZ,LOOPO
        CCF
LAB6:   LD    E,5DH               ;WAIT 0.65MSEC
LOOPP:  DEC   E
        JP    NZ,LOOPP
        LD    A,D                 ;RECALL A,
                                  ; EXCHANGE REGS &
                                  ; RETURN
        EXX
        JP    (HL)

;BYTEOUT SUBROUTINE - WRITES A BYTE ONTO TAPE
;IX = RETURN ADDRESS
;GIVEN: A IS BYTE
;USES: B
;CALLS: BITOUT
;CANNOT AFFECT:DE
BYTEOUT: LD   B,08H               ;START OF BYTEOUT
                                  ; PROGRAM
LOOPQ:  RLCA                      ;OUTPUT BYTE
        LD    HL,XX12
        JP    BITOUT
XX12:   DEC   B
        JP    NZ,LOOPQ
        AND   0FFH                ;COMPUTE AND
                                  ; OUTPUT PARITY
        JP    PO,LAB7
        CCF
LAB7:   LD    HL,XX13
        JP    BITOUT
XX13:   JP    (IX)

;BITOUT SUBROUTINE - WRITES A BIT ONTO TAPE
;HL = RETURN ADDRESS
;GIVEN: CARRY IS BIT
;USES: C
;CALLS: PULSE
;CANNOT AFFECT:DE,B,A
```

TABLE 1 (continued)

```
BITOUT:  EX   AF,AF'          ;START OF BITOUT
                              ; PROGRAM
         SCF                  ;STORE ORIGINAL
                              ; BYTE AND FLAGS
         LD   IY,XX14         ;OUTPUT 1ST PULSE
         JP   PULSE
XX14:    LD   C,6BH           ;WAIT
LOOPR:   DEC  C
         JP   NZ,LOOPR
         EX   AF,AF'          ;GET ORIGINAL FLAGS
                              ; AND BYTE
         LD   C,A             ;STORE ORIGINAL
                              ; BYTE ONLY
         EX   AF,AF'
         LD   A,C
         EX   AF,AF'
         LD   IY,XX15         ;OUTPUT 2ND PULSE
         JP   PULSE
XX15:    LD   C,6BH           ;WAIT
LOOPS:   DEC  C
         JP   NZ,LOOPS
         EX   AF,AF'          ;RECALL ORIGINAL
                              ; BYTE AND RETURN
         JP   (HL)

;SUBROUTINE PULSE - WRITE A PULSE ONTO TAPE
;IY = RETURN ADDRESS
;GIVEN: PULSE IF CARRY
;USES: C
;CANNOT AFFECT:DE,B

PULSE:   LD   A,00H           ;START OF PULSE
                              ; PROGRAM
         JP   NC,LAB8         ;SET OUT IF
                              ; REQUIRED
         OR   80H
LAB8:    LD   (CURRENT_GAME_OUT),A
                              ;OUT AND WAIT
         LD   C,24H
LOOPT:   DEC  C
         JP   NZ,LOOPT
         LD   A,00H           ;TURN OFF
         LD   (CURRENT_GAME_OUT),A
         JP   (IY)

;SUBROUTINE CONVERT - CONVERTS DATA TO 7 SEGMENT
;IY = RETURN ADDRESS
;GIVEN: A IS TO BE CONVERTED
;RESULT: A IS CONVERTED DATA

CONVERT: CP   00H             ;START OF CONVERT
                              PROGRAM
         JP   NZ,LABA
         LD   A,3FH
         JP   (IY)
LABA:    CP   01H
         JP   NZ,LABB
LABB:    LD   A,06H
         JP   (IY)
         CP   02H
         JP   NZ,LABC
         LD   A,5BH
         JP   (IY)
LABC:    CP   03H
         JP   NZ,LABD
         LD   A,4FH
         JP   (IY)
LABD:    CP   04H
         JP   NZ,LABE
         LD   A,66H
         JP   (IY)
LABE:    CP   05H
         JP   NZ,LABF
         LD   A,6DH
         JP   (IY)
LABF:    CP   06H
         JP   NZ,LABG
         LD   A,7DH
         JP   (IY)
LABG:    CP   07H
         JP   NZ,LABH
         LD   A,07H
         JP   (IY)
LABH:    CP   08H
         JP   NZ,LABI
         LD   A,7FH
         JP   (IY)
LABI:    CP   09H
         JP   NZ,LABJ
         LD   A,6FH
         JP   (IY)
LABJ:    CP   0AH
         JP   NZ,LABK
         LD   A,77H
         JP   (IY)
LABK:    CP   0BH
         JP   NZ,LABL
         LD   A,7CH
         JP   (IY)
LABL:    CP   0CH
         JP   NZ,LABM
         LD   A,39H
         JP   (IY)
LABM:    CP   0DH
         JP   NZ,LABN
         LD   A,5EH
         JP   (IY)
LABN:    CP   0EH
         JP   NZ,LABO
         LD   A,79H
         JP   (IY)
LABO:    LD   A,71H
         JP   (IY)
         END
```

has been misrecorded. If incorrect parity is detected when the computer is reading from the tape, it will stop reading, and the LAST GAME FOUND display will show a message of three horizontal bars to indicate an error.

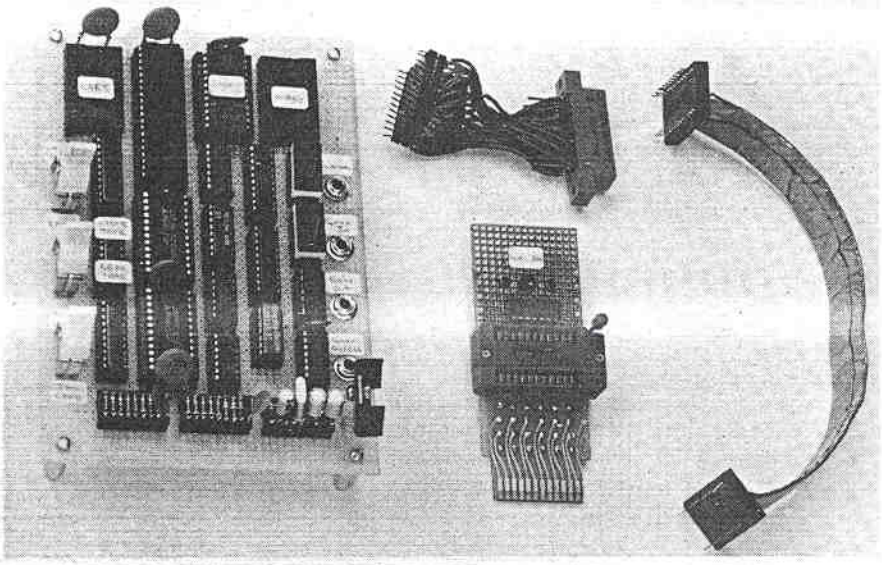
Before we go any further, we should talk a little about the memory mapping used in the game recorder. The system ROM resides from 0000H to 1FFFH. (Note that a capital "H" indicates that a number is written in hexadecimal.) The game cartridge occupies the second 8K block—2000H to 3FFFH. The game recorder's RAM is located from 4000 to 5FFFH. Cassette I/O and the displays are also memory mapped: The block from 800 0H to 9FFFH is used for the LAST GAME FOUND display and the cassette data output, while the block from A000H to BFFFH is used for the GAME SELECTED display and for the remote cassette control. The cassette data input and the switches are memory mapped from C00 0H to DFFFH. Note that two 8K blocks (6 000H–7FFFH and E000H–FFFFH) are not used.

The easiest job that our computer has to do is to read the program ROM. As it operates now, the computer can copy all 2K × 8 ROM's and 4K × 8 ROM's. As you might expect, it is possible to modify the recorder to copy 8K × 8 ROM's. Note, for example, that although an 8K block was left available for program-storage RAM, the hardware as presented has provision for only 4K.

We'll talk more about how to expand the unit to record larger programs, and show you how to build and use it, when we continue our look at the Atari game recorder next time. **R-E**

ATARI Game Recorder

GUY VACHON and DAVID A. CHAN



You can record the contents of your Atari 2600 videogame cartridges on audio cassette tape! This month, in the conclusion of this article, we'll show you how to build the game recorder and how to put it to use.

Part 3

WHEN WE LEFT OFF last time, we were describing the memory mapping technique that the game recorder uses. The last thing that we want to mention on that subject is that, for simplicity's sake, all ROM's were treated as if they were 4K × 8. That doesn't present any problems with 2K × 8 ROM's because they ignore the most-significant address bit. However, we end up with two copies of the cartridge in the 4K × 8 space—the top and bottom halves are identical. For the time being, remember that all 2K × 8 and 4K × 8 ROM's can be read by inputting to the ROM 4096 addresses (all that can be obtained from all possible combinations of 12 address bits), and saving the data patterns that the ROM returns.

One of the goals of the design of the game recorder was to keep the IC count down. Therefore, the extra memory IC's that would be required to make room for the system stack were not added. That conflicted with our desire to use subroutines (whose return addresses are usually stored in the stack). To get around that conflict, return addresses are kept in the Z80's internal registers. Thus, before a subroutine is called, the return address desired is stored in an internal register; the particular register is determined by which subroutine is to be called. The number of Z80 registers allows for up to three levels of subroutines. Besides that "trick," the software that we showed you last month is quite straightforward.

Building the game recorder

The author's prototype, shown in Fig. 6, was built on perforated construction board. Most of the connections were wire-wrapped. (Even the discrete components were wire-wrapped by first installing them in DIP headers, and then installing the header in a wire-wrap socket.) Eighth-inch phone jacks were used for cassette I/O and power connections, and 24-pin DIP sockets were used for connections to the Atari *2600* and to the game cartridge. Note that a simple power supply, whose schematic is shown in Fig. 7, was also mounted on board. The input to the supply is from a 9-volt, 500-mA DC wall transformer—similar to the transformer that the Atari *2600* itself uses.

Turning to Fig. 8, we see the connectors we need to connect the main board to the game cartridge and to the Atari *2600*. At the top left is the cartridge connector. In the author's prototype, that was basically a DIP-to-card-edge converter: One side plugs into the 24-pin DIP socket on the main board, while the other side is a 24-pin card-edge connector with standard 0.1-inch spacing. Note that to make wiring the connector easier, two 12-pin jumper headers are used to plug into the 24-pin socket. The wires from the headers then connect to the 24-pin edge connector. If you look closely at the photograph, you might note that an inverter is mounted between the socket and the card-edge connector. That's needed to invert the ENABLE line because the program ROM in the cartridge—as opposed to a 4K EPROM—is active high. The foil patterns that we'll show you shortly incorporate the inverter on the board.

Below the cartridge connector is a second board—which we'll call the *adapter board*—that is needed to connect the

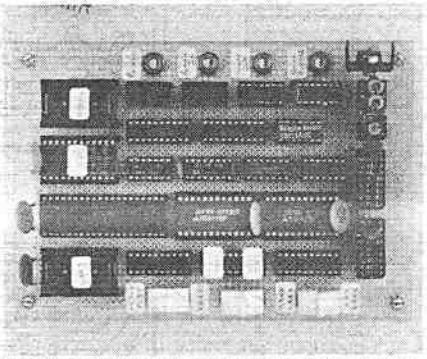


FIG. 6—THE MAIN BOARD of the author's prototype. Three SPDT switches, two 24-pin sockets, and 3½-inch phone jacks are used for input/output. The fourth jack is used for connection to a wall transformer.

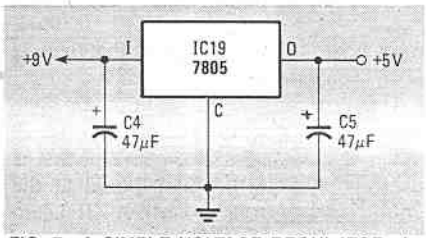


FIG. 7—A SIMPLE VOLTAGE REGULATOR circuit can be mounted on the main board.

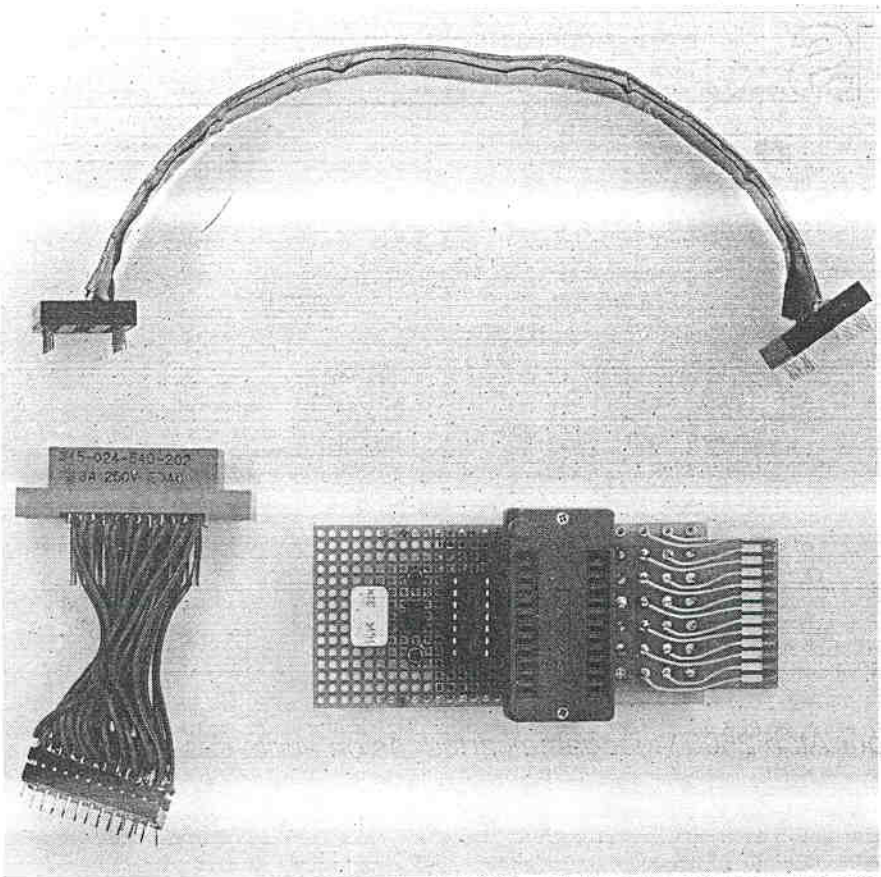


FIG. 8—TO CONNECT THE MAIN BOARD to the Atari *2600* and to the game cartridges, you need to make up special connectors.

main board to the *2600*. Note that there is also an inverter on that board.

On the right side of Fig. 8 is a ribbon cable with 24-pin connectors on each side. As you might have guessed, that is used to connect the main board to the *2600* (via the second board). A ZIF (Zero *I*nsertion *F*orce) socket is mounted on the second board for convenience. Note that the ribbon cable is shielded by copper foil. You will most likely find that shielding the cable will be necessary.

Wire-wrapping a circuit of this complexity is possible but, since wire wrapping sometimes leads to problems in troubleshooting and in mechanical integrity, a printed-circuit board is a desirable alternative. Foil patterns for the component and solder sides of the main board are shown in Figs. 9 and 10 respectively. A supplier of that board is available: See the parts list for information.

The parts-placement diagram for the main board is shown in Fig. 11. Note that there are a few differences between the PC board and the author's prototype. For example, while all the switches and jacks were mounted on the main board of the prototype, the PC board is meant to be used with panel-mounted components. Also, the inverters that were mounted on the prototype's connectors are now located on the board. Note that pull-up resistors for some of the switches are locted off-board. Those resistors, which were not shown in the schematic, are shown in Fig. 11.

You will still need an adapter board to connect main board to the *2600*, and you will have to wire up special cables to connect the main board to both the *2600* and to the game cartridge. The foil patterns for the adapter are shown in Figs. 12 and 13. Figure 14 shows the card-edge pinout of a game cartridge. That, along with Figs. 11–13 should help you wire your cables correctly.

When you build the game recorder—or any other device that uses IC's—be sure to use IC sockets. Start by installing those sockets, followed by the discrete components. Don't install any of the IC's except the voltage regulator. If you use a wall-mounted transformer, install an 1/8-inch phone jack off the board for power connections. Apply power to that jack and check for +5 volts at the appropriate IC pins. Remove power and double check the board for shorts between traces (solder bridges) or for any other potential problems. When you're confident that the board is in good shape, install the IC's, the two displays, and the relay.

Next, you'll have to install the other jacks and the switches. Since those are meant to be panel-mounted, you'll have to cut wires to the appropriate length. Once that is done, you're ready to test the unit out.

Using the game recorder

Throughout this article, we've referred to the various switches and displays that are used on the game recorder. Now it's time to tell you how to use them.

Six switches, two seven-segment displays, and three phone jacks are used for input and output to the game recorder. If you look at the photo in Fig. 6, you'll see only three switches—each single-pole, double-throw switch is used for two functions. You may want to follow the same setup: After all, you can't read and write to the cassette tape at the same time!

Let's give a brief overview of what the switches and displays do. Then we'll go

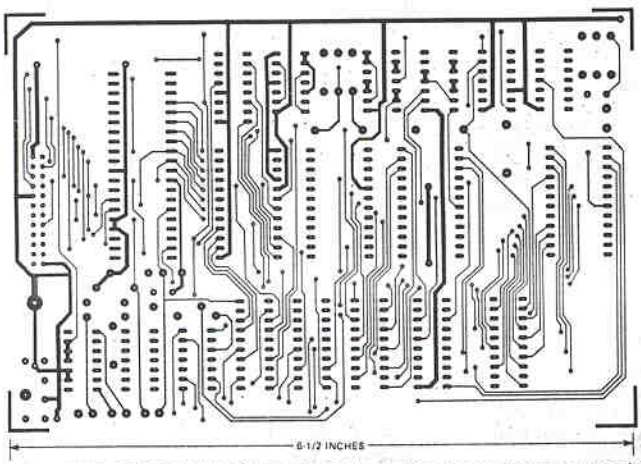


FIG. 9—THE COMPONENT SIDE of a PC board for the Atari game recorder.

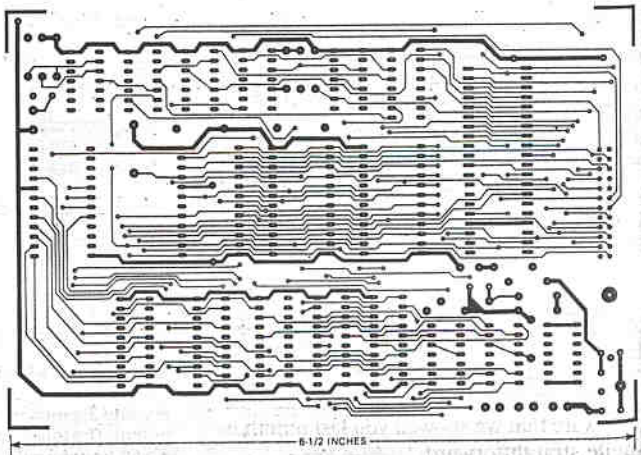


FIG. 10—THE SOLDER SIDE of the game recorder board.

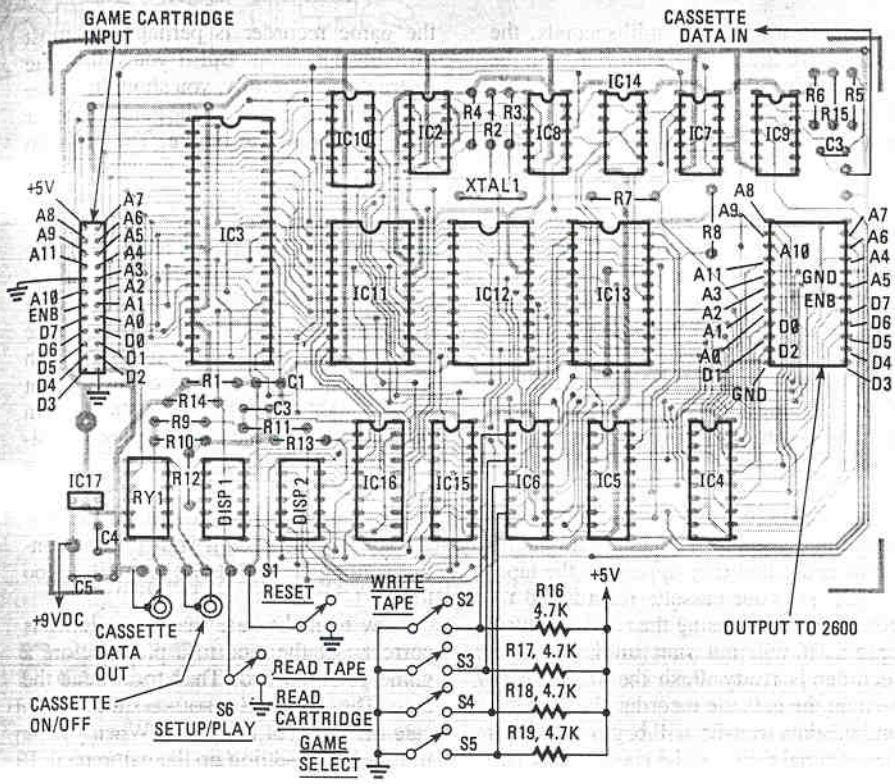


FIG. 11—PARTS-PLACEMENT DIAGRAM for the main board. Note the connector pinouts. You will have to wire up a cable to connect the main board to game cartridges and to the adapter board.

PARTS LIST

All resistors are ¼-watt, 5%, unless otherwise specified.
R1—220,000 ohms
R2, R3—1000 ohms
R4—330 ohms
R5, R7–R9—10,000 ohms
R6, R15—100,000 ohms
R10—3300 ohms
R11—47,000 ohms
R12, R13—120 ohms
R14—4700 ohms

Capacitors
C1—1 µF, 10 volts, electrolytic
C2, C3—.01 µF, ceramic disc
C4, C5—47 µF, 10 volts, electrolytic

Semiconductors
IC1, IC8, IC14—74LS00 quad 2-input NAND gate
IC2—74LS74 dual D-type flip-flop
IC3—Z80 microprocessor
IC4–IC6—74LS244 octal buffer
IC7—74LS125 quad bus buffer
IC9—LM3900 quad op-amp
IC10—74LS138 3-to-8 line decoder
IC11—2716 EPROM containing the computer's operating system
IC12, IC13—2016 2K × 8 static RAM
IC15, IC16—74LS273 octal D-type flip-flop
IC17—7805 5-volt regulator
DISP1, DISP2—MAN74A or similar seven-segment display

Other components
S1–S5—SPST momentary switch
S6—SPST toggle switch
Miscellaneous: 12/24 card-edge connector, ZIF socket, wiring harnesses to connect main board to *2600* and cartridge, etc.

An EPROM containing the game-recorder program is available for $15 postpaid from J&L Associates, 1133 Broadway Room 906, New York, NY 10010. New York residents must add sales tax.

A set of two etched, drilled, and plated-through boards are available from E²VSI, PO Box 72100, Roselle, IL 60172 for $32.50 postpaid.

into detail on each function. The first switch we'll consider is the GAME SELECT switch, S1, which is used to reset the game recorder in case of failure. The WRITE TAPE switch, S2, is used to initiate the transfer of data from the game recorder's RAM to a cassette tape. Switch S3, READ TAPE, does just the opposite: It initiates data transfer from cassette tape to the game recorder's RAM. The READ CARTRIDGE switch, S4, initiates the transfer of data from the game cartridge to the game recorder's RAM. Switch S5, GAME SELECT, is used to select the name of the game that you want to save on tape, or the name of the game you want to find on a tape. You have 16 choices for a name: the hexadecimal digits Ø–F. The final switch, S6, is the SETUP/PLAY switch which is used to put the game recorder in the mode to play a game on the *2600*.

The first of the two displays, GAME SELECT, shows the name of the game that you want to save or the one you are trying to find on a tape. The name is selected by the GAME SELECT switch. The LAST GAME FOUND display is used to indicate the name of the game that the recorder is "listening to" on the tape. We'll see that the decimal points of those displays serve another important function.

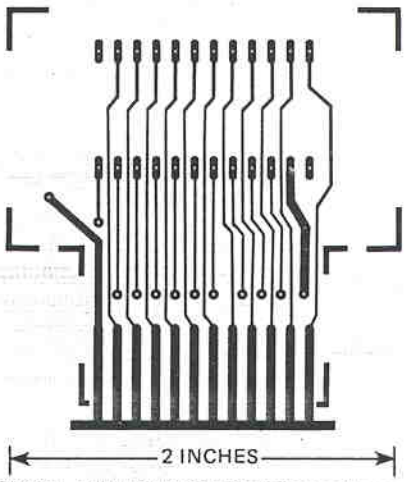


FIG. 12—THIS ADAPTER BOARD is used to connect the main board to the Atari *2600*. We recommend that you use a DIP socket for convenience.

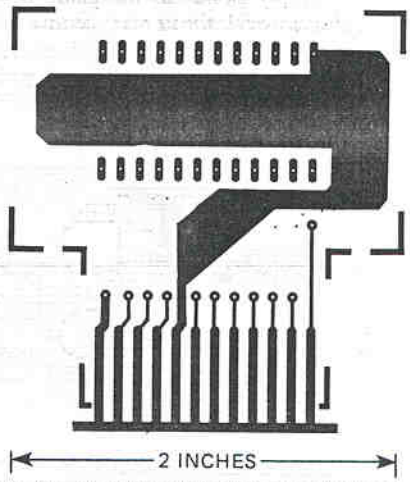


FIG. 13—THE FOIL SIDE OF the adapter board.

There is one cassette input to the game recorder (CASSETTE DATA IN) and two cassette outputs (CASSETTE DATA OUT and CASSETTE ON/OFF). Note that the schematic does not show the CASSETTE ON/OFF output. Instead, the output of IC7-c is labeled "TO CASSETTE-CONTROL RELAY." Although the relay is not shown on the schematic, the board has provision for a DIP mounted relay that can be controlled by the output of IC7-c.

Before you use your game recorder, keep the following notes of caution in mind: **NEVER** plug a game cartridge in while the game recorder is on. The *2600* should be turned **OFF** before the game

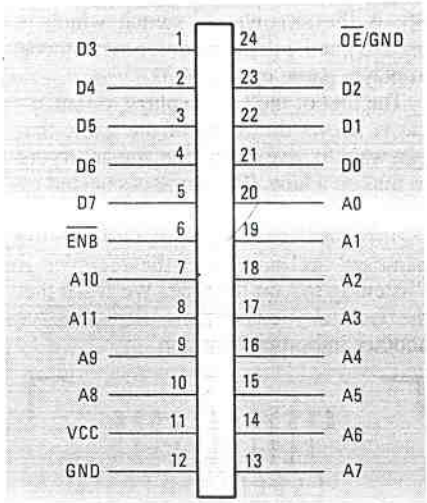


FIG. 14—GAME CARTRIDGE PINOUT. Pin 1 is at the lower left of the front (label side) of the cartridge.

recorder is connected. **NEVER** turn the *2600* on while the game recorder is not in the PLAY position.

Reading a cartridge

Reading a game cartridge is perhaps the easiest task that the game recorder has. It's also an easy function for you to initiate. First make sure that the game recorder's power is off and that it is properly hooked up to the *2600*. Then make sure that the SETUP/PLAY switch is in its SETUP position. Apply power to the game recorder and close S4, the READ CARTRIDGE switch. In a matter of milliseconds, the contents of the game cartridge will be transferred to the game recorder's RAM. If you want to verify that the copy is correct, you can play the game.

Writing a game to tape

Once you have a game stored in RAM, you can transfer it to cassette tape. First make sure that the game is set up as before. If you're using a cassette control relay, attach the cassette output to the REMOTE jack of the cassette recorder. Select the name of the game (a hex digit from Ø-F) by pushing the GAME SELECT switch. Each time you push it, the GAME SELECT display will increment by one. That will place a "label" on the tape that the game recorder will be able to find at another time. It's a good idea to name your games in the order that they appear on the tape.

Next set your cassette recorder to record. (If you are using the remote control option, it will not start until the game recorder is ready. Push the WRITE TAPE switch; the cassette recorder should start, and the data transfer will begin. Note that the decimal point of the GAME SELECTED display will light. After the game has been transferred, the decimal point will again go dark. That's your signal to stop your cassette recorder if you're not using the remote control option.

Reading a tape

Loading a game from cassette tape to the game recorder is perhaps the most difficult operation. But if you follow the instructions carefully, you shouldn't have any problems. Hook up the game recorder in its setup mode as before. The first step is to set the volume level. Once you learn what the proper level is, you won't have to repeat this step every time.

Set your tape to play a recorded game (which you can recognize by the high- and low-pitched tones). Then turn the volume down and connect the cassette's earphone jack to the CASSETTE-DATA IN input. Push the READ TAPE button; the decimal point of the GAME SELECT display should turn on. Now turn up the cassette player's volume until the decimal point of the LAST GAME FOUND display just turns on. Then turn the volume up just a slight bit more—about a half number, if your volume control is numbered. But don't turn it up too loud.

Now to make sure that the volume is correct, set the tape to a place before a game you recorded. Then try to read the tape. The LAST GAME FOUND should indicate the name of the game. When you're finished with setting up the game recorder, press the RESET button.

Now try to load back a game you recorded by selecting its name, setting up the recorder, and hitting the read tape button. The LAST GAME FOUND display will indicate the name of each game the game recorder finds. When it finds the selected game, it will read it and stop the cassette player when it's finished (if the remote option is used).

Playing a game

Now that you have a game in the game recorder's memory—either from a tape or from a game cartridge—you can play it on the *2600*. Just move the SETUP/PLAY switch to the PLAY position and turn the *2600* on. Your game should be ready to play. Be sure to turn the *2600* off before moving the SETUP/PLAY switch back to SETUP.

Expanding the game recorder

As a final note, we should note that many new games for the Atari *2600* are 8K long. The basic ideas of this game recorder can be used to record those games on cassette tape. However, you will have to make both hardware and software modifications. We won't go into detail on how those modifications are made, but we will give you a head start.

Figure 15 shows one way of adding to the the address decoder to obtain bankswitching ability. The software would have to be written to turn the memory-mapped bankswitching mechanism on and off (by setting and resetting a flip-flop, for example). The software should include some way to detect and record the ROM size. The last two bits of the header could be used for that purpose. **R-E**

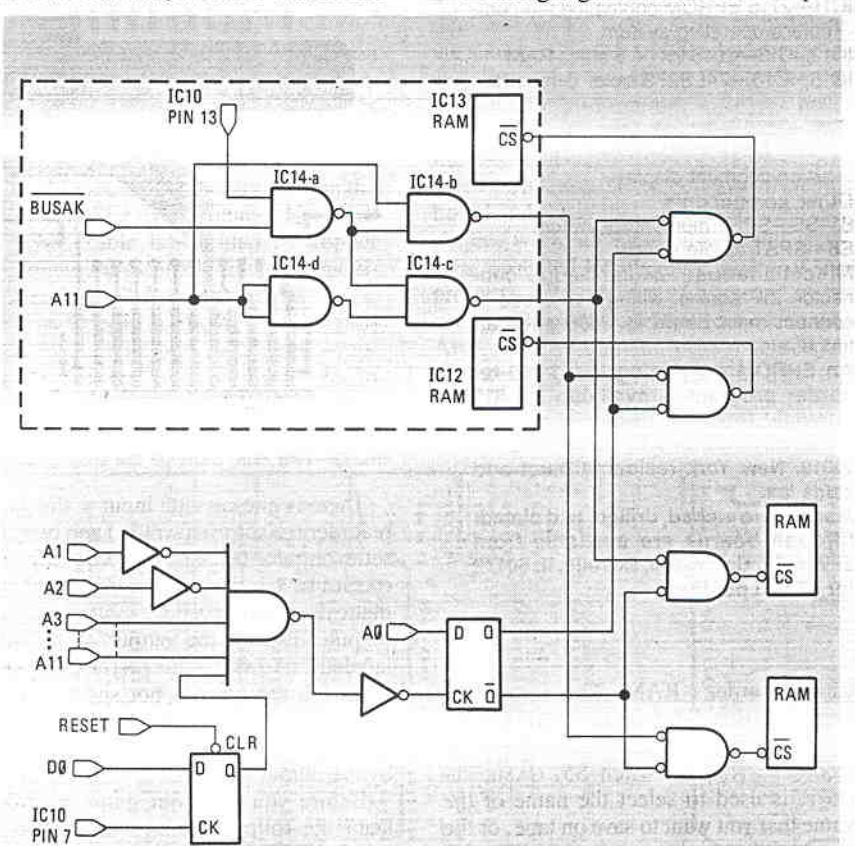


FIG. 15—EXPANDING THE RECORDER to record larger games is possible. The software will have to be changed, as will the address decoder. Here is a possible decoder scheme to use for bankswitching.